Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Смоленский государственный университет»

Кафедра прикладной математики и информатики

«Утверждаю»
Проректор по учебно-
методической работе
Ю.А. Устименко
«23» июня 2022 г.

Рабочая программа дисциплины
Б1.В.ДВ.01.02 Машинная графика

Направление подготовки: **09.03.03 Прикладная информатика**
Направленность (профиль): **Информационные системы организаций и предприятий**
Форма обучения: заочная
Курс – 2
Семестр – 3, 4
Всего зачетных единиц – 3, часов – 108
Форма отчетности: зачет – 4 семестр

Программу разработал
кандидат педагогических наук, доцент Киселева О.М.

Одобрена на заседании кафедры
«16» июня 2022 г., протокол № 10

Заведующий кафедрой _____ С.В. Козлов

Смоленск
2022

1. Место дисциплины в структуре ОП

Дисциплина Б1.В.ДВ.01.02 «Машинная графика» относится к части, формируемой участниками образовательных отношений Блока 1, дисциплины (модули) по выбору 1 (ДВ.1).

При изучении данной дисциплины необходимы компетенции студентов, сформированные при изучении таких дисциплин, как «Основы информатики», «Языки и методы программирования». Курс построен так, чтобы углубить и расширить тот объем знаний по разделам, связанным с применением методов программирования для решения задач машинной графики.

Изучение курса основано на традиционных методах высшей школы, тесной взаимосвязи со смежными курсами, а также на использовании современных систем компьютерной обработки изображений.

2. Планируемые результаты обучения по дисциплине

Компетенция	Индикаторы достижения
ПК-1- Способен проводить обследование организаций, выявлять информационные потребности пользователей, собирать детальную информацию, формировать требования к автоматизированной информационной системе (ERP-системе)	**Знать**: методику проведения обследования организаций с целью выявления информационных потребностей пользователей; требования, предъявляемые к логистической информационной системе; возможности типовых ИС, архитектуру, устройство и функционирование вычислительных сетей, коммуникационное оборудование и сетевые протоколы, теорию баз данных и основы программирования; основы бухгалтерского учета, управления торговлей, поставками, запасами, управления персоналом, управления организацией, экономической теории. **Уметь**: выявлять информационные потребности пользователей, формулировать требования к логистической информационной системе, осуществлять сбор детальной информации для формализации требований пользователей заказчика. **Владеть:** методами, способами и инструментами выявления информационных потребностей пользователей, методикой обследования организации, навыками по информированию заказчика о возможностях типовых ИС..
ПК-3 - Способен создавать прототипы автоматизированных информационных систем (ERP-систем), разрабатывать программный код информационной системы и баз данных информационной системы для управления бизнес-процессами, создавать прикладное программное обеспечение	**Знать:** современные языки программирования, их синтаксис, языки программирования и работы с базами данных, теорию баз данных, инструменты и методы тестирования характеристик ИС и прототипирования пользовательского интерфейса, возможности типовой ИС, ее устройство и функционирование, основы современных операционных систем, систем управления базами данных. **Уметь**: кодировать на языках программирования, тестировать результаты прототипирования, тестировать

	разрабатываемую ИС (модульное, интеграционное тестирование), обнаруживать и устранять несоответствия и дефекты. **Владеть**: навыками по созданию прикладного программного обеспечения, разработке прототипов ИС, разработке кода ИС и баз даны ИС, тестирования ИС, устранения обнаруженных несоответствий и дефектов.

3. Содержание дисциплины

1. Теория цвета и обработка изображений, фильтры. Цвет и цветовые модели. Компьютерная графика. Обработка изображений. Компьютерное зрение. Визуализация. Растровая и векторная графика. Понятие растра. Представление цвета в компьютерной графике. Аддитивная модель. Законы Г. Грассмана: закон трехмерности, закон непрерывности и закон аддитивности. Субтрактивные цветовые модели CMY и CMYK. Проблема разложения монохромного цвета. Цветовые модели CIE: XYZ, диаграмма цветности CIE, L*u*v*, L*a*b*. Преобразования между CIE XYZ и RGB. Цветовые модели CIE L*u*v* и CIE L*a*b*. Цветовые модели пользователя. Модели HSV, HSB. Алгоритмы преобразования из RGB в HSV и обратно. Цветовые модели, разделяющие яркость и цвет: Y**. Цветовые модели YUV, YPbPr и YCbCr. Цветовая модель YIQ.

Получение цифрового изображения. Причины потери качества изображения. Гистограмма. Коррекция яркости/контраста изображения. Линейная коррекция. Нелинейная коррекция. Гамма-коррекция. Компенсация разности освещения. Выравнивание освещения. Цветовая коррекция изображений. Гипотеза «Серый мир». Гипотеза «Идеальный отражатель». Растяжение контрастности каналов. Коррекция с опорным цветом. Статистическая цветокоррекция. Борьба с шумом. Шум в бинарных изображениях. Подавление и устранение шума. Операции матморфологии: расширение и сужение. Свойства морфологических операций. Дискретные операции морфологии. Алгоритмы морфологического расширения и сужения. Операции открытия и закрытия. Шум в бинарных изображениях с дефектами объектов. Устранение шума в полутоновых и цветных изображениях. Причины и примеры шума изображения. Операция «свертка». Усреднение. Подавление и устранение шума. Медианный фильтр. Фильтр Гаусса. Преобразование Фурье. Адаптивные фильтры. «Продвинутые» фильтры. Примеры шумоподавления. Выделение контуров. Операция оконтуривания объекта. Выделение точек контура. Градиент. Приближения (маски) Робертса, Превитта и Собеля. Спецэффекты. Тиснение. Цифровой негатив. Светящиеся края. Перенос/поворот. «Волны». Эффект «стекла».

2. Параметрические полиномиальные кривые и поверхности. **Фракталы**. **Метод систем итеративных функций.** Моделирование кривых. Однородные координаты. Параметрические кривые. Интерактивное конструирование кривых. Кривая Безье. Многочлены Бернштейна. Свойства кривых Безье. Задача интерполяции. Задача сглаживания. Сплайны. Сплайны Безье. B-сплайны и NURBS. Математические выражения для кривой и поверхности. Аппроксимация функции двух переменных. Аппроксимация произвольных функций с помощью NURBS. Кривые и поверхности NURBS. Множество стыковочных функций. Рекурсия Кокса-де Бура. Поведение NURBS при наличии коллинеарных вершин. Построение сплайна без определения узловых производных. Построение замкнутого сплайна без определения узловых производных. Сетки узлов. Бета-сплайны. Уравнение бета-сплайна. Свойства бета-сплайна.

История появления фракталов. Геометрические фракталы. Кривые Коха. Алгоритм рисования кривых и снежинок Коха. Простейшие алгоритмы рисования фрактальных кривых. Метод L – систем. Кривая Госпера. Квадратный остров Коха. Наконечник Серпинского. Кривая Гильберта. Разрешение ветвления и фрактальные деревья. Системы итерируемых функций. Экспериментальный копир. Аттрактор. Построение кривой Коха и «дракона» Хартера-Хентуэя на основе IFS. Общие алгоритмы рисования k-итерации IFS. Алгебраические фракталы.

Множество Мандельброта. Случайные фракталы. Спектральная плотность фрактальной кривой. Фрактальные поверхности. Фрактальное сжатие изображений.

3. Базовые растровые алгоритмы основные алгоритмы вычислительной геометрии. Координатный метод в компьютерной графике. Локальные модели освещения. Растеризация линий. Прямое вычисление. Цифровой дифференциальный анализатор (ЦДА, DDA). Инкрементные алгоритмы. Алгоритмы Брезенхэма для отрезка и окружности. Модификация алгоритма Брезенхэма со сглаживанием границы. Алгоритмы закрашивания. Алгоритмы вывода фигур. Алгоритмы закрашивания до цвета границы: простейший рекурсивный алгоритм, закрашивание линиями. Построчное заполнение контура полигона, заданного списком вершин. Правило подсчета числа пересечений ребер с горизонталью. Построчное заполнение полигона, заданного списком вершин: возможности оптимизации. Стиль линии. Перо. Кисть. Отсечение отрезка – алгоритм Кохена (Коэна)-Сазерленда. Классификация положения точки относительно отрезка (справа, слева, спереди, сзади). Расстояние от точки до прямой (плоскости). Пересечение двух отрезков (плоскостей). Проверка принадлежности точки полигону. Вычисление площади полигона. Построение выпуклой оболочки множества точек (заворачивание подарка и др. алгоритмы). Построение звездчатого полигона (ядра полигона: полигонализация набора S вершин – все вершины должны быть видны из вершины s0, принадлежащей ядру полигона). Пересечение выпуклых полигонов (алгоритм Сазерленда-Ходжмана). Построение триангуляции Делоне.

Векторные полигональные модели 3D объектов. Однородные координаты. Преобразование объектов сцены. Повороты и параллельный перенос в 3D. Преобразование нормалей при деформации поверхности. Иерархическая систематизация проекций. Аксонометрические проекции. Ортогональное проецирование в OpenGL и канонический видимый объем (CVV). Перспективные проекции. Геометрическая интерпретация одноточечной перспективы. Методы создания перспективных видов. Вращение в сочетании с одноточечной перспективой. Фотография и перспективные преобразования. Стереографические проекции и виртуальная реальность. Стереографика: элементы технологии. Серый анаглиф. Цветной анаглиф. Полуцветной анаглиф. Восстановление трехмерных объектов по перспективным проекциям. Захват движения.

Метод трассировки лучей. Материалы. Закрашивание поверхностей. Свет и материя. Источники света. Цвет излучения. Фоновое освещение. Точечный источник света. Прожекторы. Удаленный источник света. Модель Фонга для отражения. Отражение фонового света. Диффузное отражение. Зеркальное отражение. Вычисление векторов. Нормаль к поверхности. Угол отражения. Вектор половинного направления. Преломление света. Закрашивание многоугольников. Плоское закрашивание. Интерполяционное закрашивание и закрашивание по методу Гуро. Закрашивание по методу Фонга. Элементы глобального освещения сцены. Алгоритмы локального освещения. Метод растеризации. Метод трассировки лучей. Проблема реального времени и подходы к упрощению расчета освещенности. Примеры синтезированных изображений. Классическая трассировка лучей. Трассировка лучей (Ray tracing). Прямая и обратная трассировка. Ограничения методов. Генерация первичного луча. Расчет прямого освещения. Расчет вторичного освещения. Учет отражения света. Учет преломления света. Процесс трассировки. Рекурсивная процедура трассировки. Модель Уиттеда.

4. Базовое программное обеспечение 3D-графики. 3D-сцена и графический конвейер. Геометрическая стадия. Стадия рендеринга. Обобщенная структура 3D-акселератора (видеокарты). Геометрический процессор акселератора. Структура данных о полигональной модели. Шейдеры. Архитектура шейдеров. Вершинные шейдеры. Пиксельные шейдеры. Современные акселераторы (видеокарты). Microsoft HLSL (High Level Shading Language). Шейдеры в OpenGL. Язык GLSL. Язык OpenGL. MS DirectX.

Текстурирование. Удаление невидимых элементов и оптимизация. Тени. Модель наложения текстуры на поверхность. Быстрый метод отображения текстур. Алгоритм быстрого наложения текстуры на треугольник. Mipmapping. Построение mip уровней (оценка по памяти). Методы расчета LOD. 3.3.Методы формирования значения текстуры. Учет перспективы при отображении: примеры. Эффективная реализация алгоритма с учетом перспективы. Анализ

методов сглаживания на основе super-sampling. Артефакты и методы их сглаживания. Практическая реализация сглаживания. Методы избыточной выборки: RGSS и OGSS. Осуществление OGSS. Удаление невидимых линий и поверхностей. Основные виды проецирования. Отсечение нелицевых граней. Удаление невидимых линий. Алгоритм Робертса. Алгоритм Аппеля. Удаление невидимых граней. Метод z-буфера. Метод сортировки по глубине. Метод построчного сканирования. Оптимизация вычислений. Когерентность. Разбиение пространства. Метод двоичного разбиения пространства. Оптимизация метода построчного сканирования для игр. Алгоритм Варнака. Метод оболочек. Иерархические структуры. Аппарат буфера трафарета в OpenGL. Аппарат буфера глубины в OpenGL. Создание теней при помощи метода теневых объемов. Построение теней при помощи теневых карт (shadow maps).

4. Тематический план

№ п/п	Разделы и темы	Всего часов	Формы занятий (в соответствии с учебным планом)				
			лекции	семинары	практические занятия	лабораторные занятия	самостоятельная работа
Семестр 3							
1	Теория цвета и обработка изображений, фильтры.	25	–	-	–	4	21
2	Параметрические полиномиальные кривые и поверхности. Фракталы. Метод систем итеративных функций	34	–	-	–	12	22
3	Базовые растровые алгоритмы основные алгоритмы вычислительной геометрии. Координатный метод в компьютерной графике. Локальные модели освещения.	37	–	-	–	16	21
	Итого	72				8	64
Семестр 4							
4	Базовое программное обеспечени	32	–	-	–	2	30

	е 3D-графики.						
	Зачет	4					4
	Итого	36				2	34
ИТОГО		108	–	-	–	10	98

5. Виды образовательной деятельности

Лабораторные работы

Лабораторное занятие 1

Задания

1. Разработать простейшее графическое приложение на C#, которое выводит надпись «Welcome, Windows Forms!» и кнопку «Выход» (рис. 1).

2. Добавьте на главную форму приложения еще одну кнопку и добейтесь того, чтобы при щелчке на данной кнопке надпись «Welcome, Windows Forms!» меняла свое положение и цвет. Для задания положения и цвета служат свойства ***Location*** и ***BackColor***.

3. Разработать простейшее графическое приложение на C# с обработчиками событий, которое демонстрирует окно с единственной кнопкой, которую невозможно нажать, т.к. она «убегает» от указателя мыши. В качестве фона выберите графическое изображение (рис. 2).

4. Усложните поведение кнопки при наведении на нее указателя мыши. Например, наряду с изменением положения задайте случайный цвет кнопки (с помощью генератора случайных чисел) и отобразите в произвольном месте окна надпись «Не нажал!». Для генерирования случайных целых и вещественных чисел используется стандартный класс System.Random.

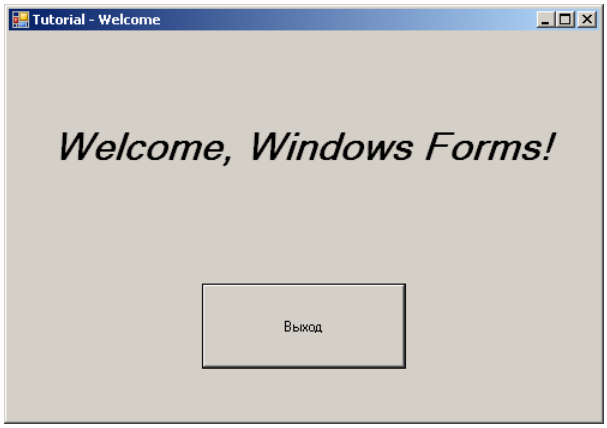


Рис. 1

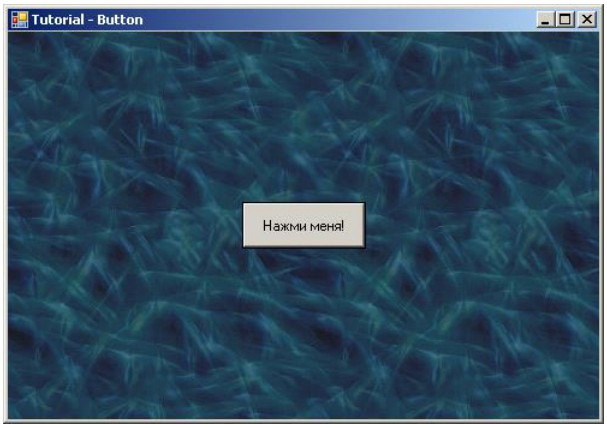


Рис. 2

1. Разработать простейшее графическое приложение на C#, которое выводит на форму один из графических примитивов (см. лекцию).

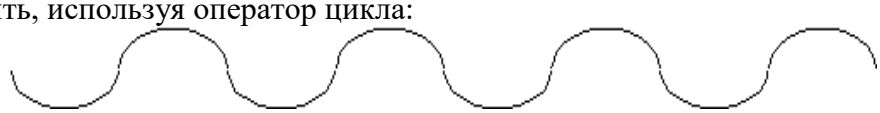
2. Выполнить индивидуальное задание лабораторной работы в соответствии с вариантом, определенным преподавателем (см. ниже приложение). Выполнить два варианта – один с 1 по 5 в лаборатории, второй – с 6 по 10 индивидуально дома.

Приложение. Лабораторная работа 2. Построение графических изображений

Вариант 1.

Изобразить на экране конус, выделив пунктиром невидимые линии. Провести высоту конуса и радиус основания. Вывести на экран формулу для вычисления объема конуса.

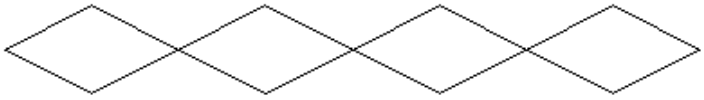
Изобразить, используя оператор цикла:



Вариант 2.

Изобразить на экране цилиндр, выделив пунктиром невидимые линии. Вывести на экран формулу для вычисления объема цилиндра.

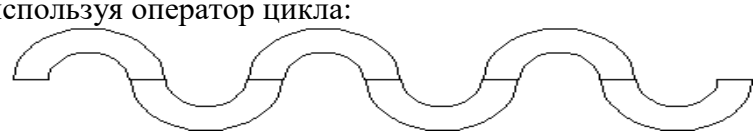
Изобразить, используя оператор цикла:



Вариант 3.

Изобразить на экране конус, выделив пунктиром невидимые линии. Вывести на экран формулу для вычисления объема конуса.

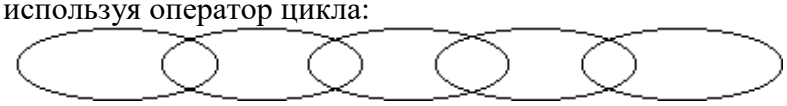
Изобразить, используя оператор цикла:



Вариант 4.

Изобразить на экране куб, выделив пунктиром невидимые линии. Вывести на экран формулу для вычисления площади полной поверхности куба.

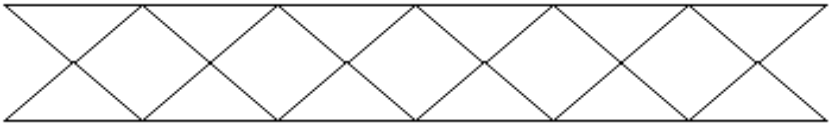
Изобразить, используя оператор цикла:



Вариант 5.

Вывести на экран произвольный отрезок и два треугольника, симметричные относительно него.

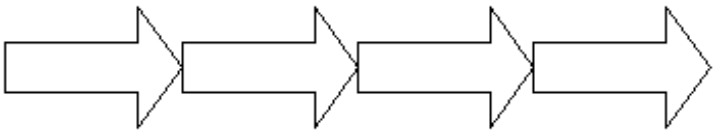
Изобразить, используя оператор цикла:



Вариант 6.

Изобразить на экране дисплея светофор.

Изобразить, используя оператор цикла:



Вариант 7.

Изобразить на экране трапецию, выделив пунктиром ее высоту. Вывести на экран формулу для вычисления площади трапеции.

Изобразить, используя оператор цикла:

Вариант 8.

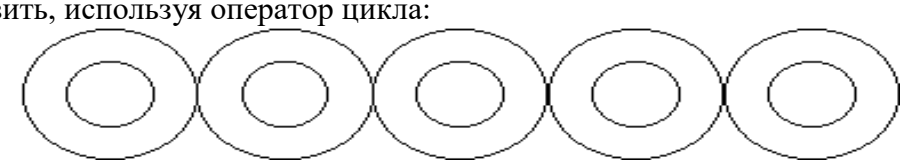
Изобразить на экране параллелограм, выделив пунктиром его высоту. Вывести на экран формулу для вычисления площади параллелограмма.

Изобразить, используя оператор цикла:

Вариант 9.

Изобразить на экране ромб, выделив пунктиром его диагонали.

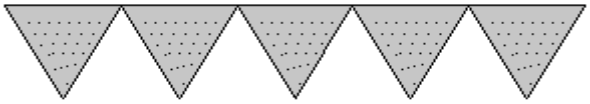
Изобразить, используя оператор цикла:



Вариант 10.

Изобразить на экране треугольник, выделив пунктиром его высоту. Вывести на экран формулу для вычисления площади треугольника.

Изобразить, используя оператор цикла:



Лабораторное занятие 2

Задания

1. Нарисовать картинку, изображающую:

- деревенский дом и окружающую природу (озеро, деревья);
- городской дом и шоссе с автомобилем (или другим транспортом);
- снеговика в зимнем лесу (ели, деревья без листьев);
- море, берег, парусник.

Разработать и реализовать индивидуально подобный проект дома.

2. Разработать программу рисования графических примитивов. Выбор графического примитива осуществить с помощью элемента управления ListBox, используя оператор выбора Case.

3. Добавить в программу рисования графических примитивов возможность изменения цвета фона формы BackColor с помощью перебора стандартных констант цвета. Также предусмотреть функцию выбора цвета для рисуемого графического примитива. Выбор организовать при помощи элементов управления ListBox.

1. Используя готовые картинки в формате BMP построить:

- рощу лиственных и хвойных деревьев;
- луг из разноцветных цветов;
- штабель из ящиков и коробок;
- небо в облаках.

Разработать и реализовать индивидуально подобный проект дома.

2. В окружении предметов переднего и заднего планов имитировать движение:

- автомобиля;
- самолёта;
- лодки;
- паровоза.

Разработать и реализовать индивидуально подобный проект дома.

1. Разработайте приложение для просмотра графических файлов (библиотека GDI+ поддерживает практически все распространенные форматы), которое демонстрирует использование основных компонент графического интерфейса (в частности, компоненты System.Windows.Forms.PictureBox) и работу с диалогами (рис. 3).

2. Поэкспериментируйте со свойствами элемента управления System.Windows.Forms.PictureBox. Реализуйте возможность сохранения загруженного изображения в файл. Для сохранения в файл используется стандартный класс

System.Windows.Forms.SaveFileDialog. Загруженное изображение хранится в поле Image компоненты System.Windows.Forms.PictureBox.

3. С помощью использования элемента PictureBox предусмотрите отображение «большого» растрового файла (по геометрическим размерам) с возможностью прокрутки (рис. 3а).

Рис. 3а Вывод изображения с возможностью прокрутки

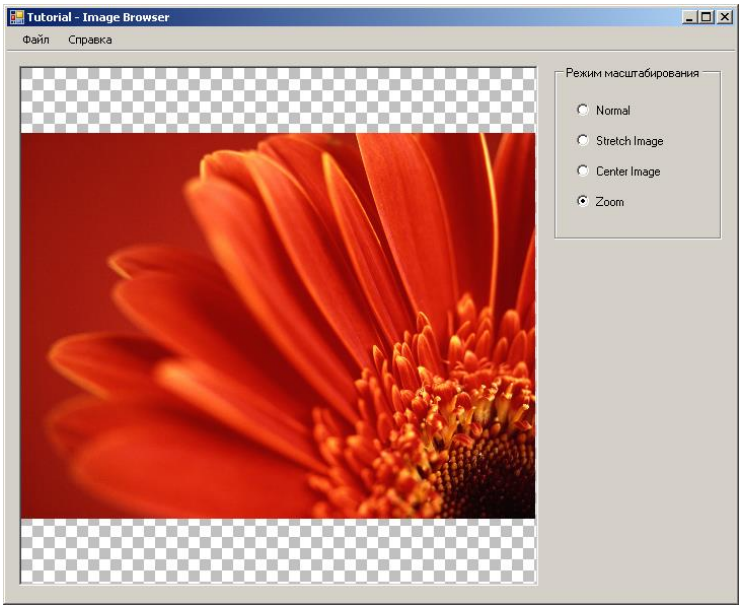


Рис. 3

Лабораторное занятие 3

Задания

1. Разработать простейшее приложение GDI+, которое выводит строку приветствия «Welcome, GDI+!», закрашенную плавно меняющимся цветом (линейный градиент) на фоне, загружаемом из файла (рис. 4).

Рис. 4

2. Поэкспериментируйте с параметрами вывода строки текста: измените используемый шрифт (переменная типа System.Drawing.Font), фоновое изображение, а также кисть, используемую для заливки строки. В данном приложении используется кисть для создания линейного градиента на основе двух цветов – System.Drawing.Drawing2D.LinearGradientBrush. Однако в пространстве имен System.Drawing.Drawing2D можно найти и другие интересные кисти, такие как кисть для создания простых узоров (HatchBrush) или для градиентной заливки вдоль произвольной кривой (PathGradientBrush). Более простые кисти находятся в пространстве имен System.Drawing, такие как кисть для сплошной заливки (SolidBrush) или кисть для заливки рисунком, который можно загрузить из файла (TextureBrush).

1. Разработать приложение GDI+, которое демонстрирует основные инструменты и методы GDI+ на примере рисования аналоговых часов. Кроме того, данный пример демонстрирует использование невизуальной компоненты System.Timer.

Рис. 5

2. Переделайте приложение из предыдущего пункта таким образом, чтобы на экране отображались цифровые и аналоговые часы (например: 15:00:30). Для этого вам потребуется выводить средствами GDI+ текстовые строки (соответствующие материалы для программного кода можно найти в лекции и у преподавателя).

Составить параметрическое уравнение и построить траекторию движения крайней точки диска, если диск катится по:

- оси Ox;
- оси Oy;
- внешней части единичной окружности;

- внутренней части единичной окружности.

Задание

Разработать проект, например «Поле чудес» или «Построение в линзах».

Лабораторное занятие 4

1. Разработать простейшее приложение GDI+, которое демонстрирует работу с объемным текстом. Нарисовать на форме вдавленный (рис. 6), выпуклый (рис. 7), контурный текст (рис. 8). Предусмотреть три кнопки для переключения между видами объемного текста.

Для того чтобы текст на форме выглядел более красивым, его стараются сделать объемным. Существует два вида объемности – вдавленный и выпуклый текст. Для достижения подобных эффектов достаточно сместить второй такой же текст относительно первого, но отобразить его другим цветом, который будет имитировать тень. Этот способ широко используется во многих графических редакторах. Для достижения результата достаточно вызвать два раза метод DrawString с разными смещениями.

Текст, выводимый на экран, по умолчанию заполнен каким-то цветом. Но можно вывести только контур символов при помощи метода DrawPath класса GraphicsPath. Данный класс позволяет создавать различные контуры из фигур и текстов.

2. Разработать простейшее приложение GDI+, демонстрирующее работу с текстом, который повернут на заданный угол (рис. 9):

- получить текст в его зеркальном отражении;
- повернуть текст на заданный угол;
- повернуть текст относительно заданной точки (фигуры) – эффект «солнышка».

Для поворота текста можно использовать метод RotateTransform.

3. Добавьте на форму кнопку для иллюстрации бегущей градиентной строки.

Рис. 6. Вдавленный текст

Рис. 7. Выпуклый текст

Рис. 8. Контурный текст

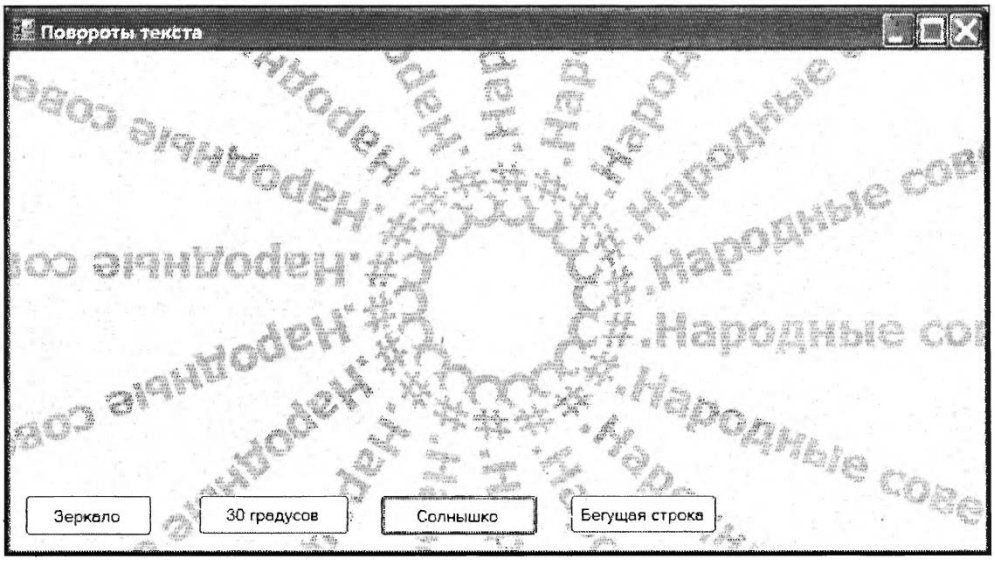


Рис. 9. Повороты текста

Программные решения для лабораторной работы № 9 (см. Котов).

1. Разработать простейшее приложение GDI+, которое отображает диаграммы изменения курсов доллара и евро (рис. 10). Числовые данные загружать из файла в массив. При разработке проекта использовать методы DrawString, DrawRectangle и FillRectangle.

2. Разработать простейшее приложение GDI+, которое отображает графики изменения курса доллара и евро (рис. 11). Числовые данные загружать из файла в массив. При разработке проекта использовать методы DrawString, DrawRectangle и FillRectangle.

3. Разработать простейшее приложение GDI+, которое отображает данные социологического опроса в идее круговой диаграммы (рис. 12). Исходные данные (вопрос, варианты ответа и количество ответов) загружаются из файла. Программа должна обработать входные данные (вычислить долю каждой категории в общей сумме) и построить диаграмму.

Рис. 10. Проект «Диаграмма изменения курса валюты»

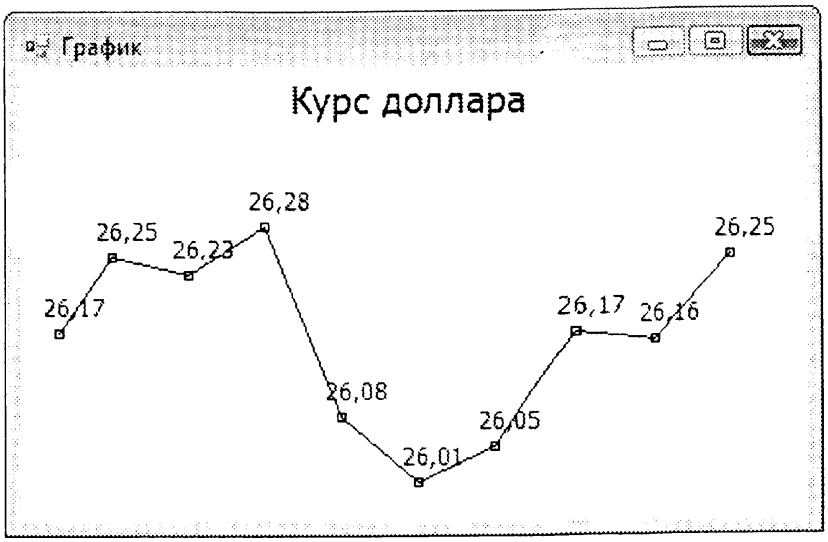


Рис. 11. Проект «График изменения курса валюты»

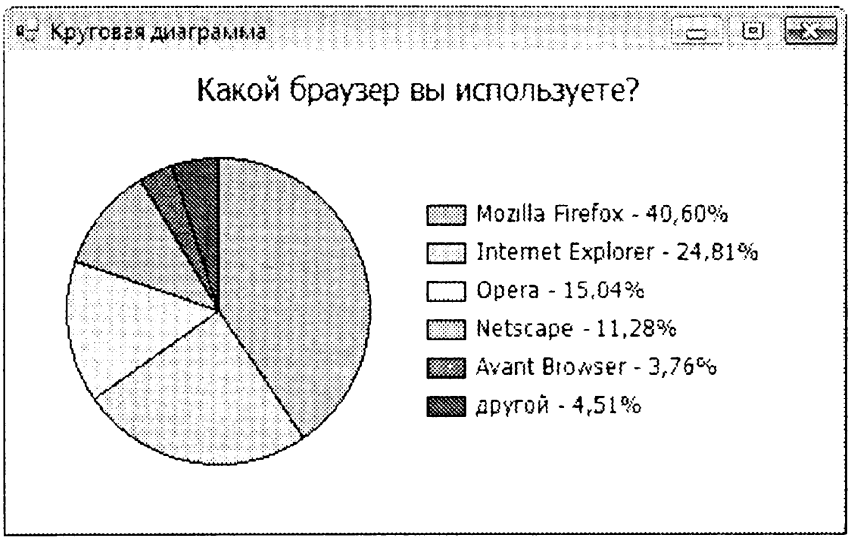


Рис. 12. Проект «Круговая диаграмма данных социологического опроса»

Программные решения для лабораторной работы № 10

- задания № 1, 2 и 4 (см. Культин – с. 90).

1. Даны сведения об объемах продаж за пять месяцев. Необходимо наглядно представить эти данные в виде гистограммы. Требуется разработать простейшее приложение GDI+, которое демонстрирует построение диаграммы по табличным данным с использованием элемента Chart.

Элемент Chart предназначен для вывода в экранную форму графика (диаграммы). При этом график удобно строить по табличным данным, представленным в виде объекта класса DataTable. Для отображения табличных данных используйте сетку данных DataGridView. Объект DataTable используют в качестве исходных данных и для сетки DataGridView, и для диаграммы Chart.

В таблице DataTable определить ее схему, задав две колонки «Месяц» и «Объем продаж». Заполнить таблицу по ее строкам (рядам), используя метод Add. Заполненную пятью строками таблицу указать в виде источника данных для элементов Chart и DataGridView. После чего оформить внешний вид гистограммы (рис. 13). Также предусмотреть в программе изменение внешнего вида столбцов гистограммы по щелчку мыши на цилиндры.

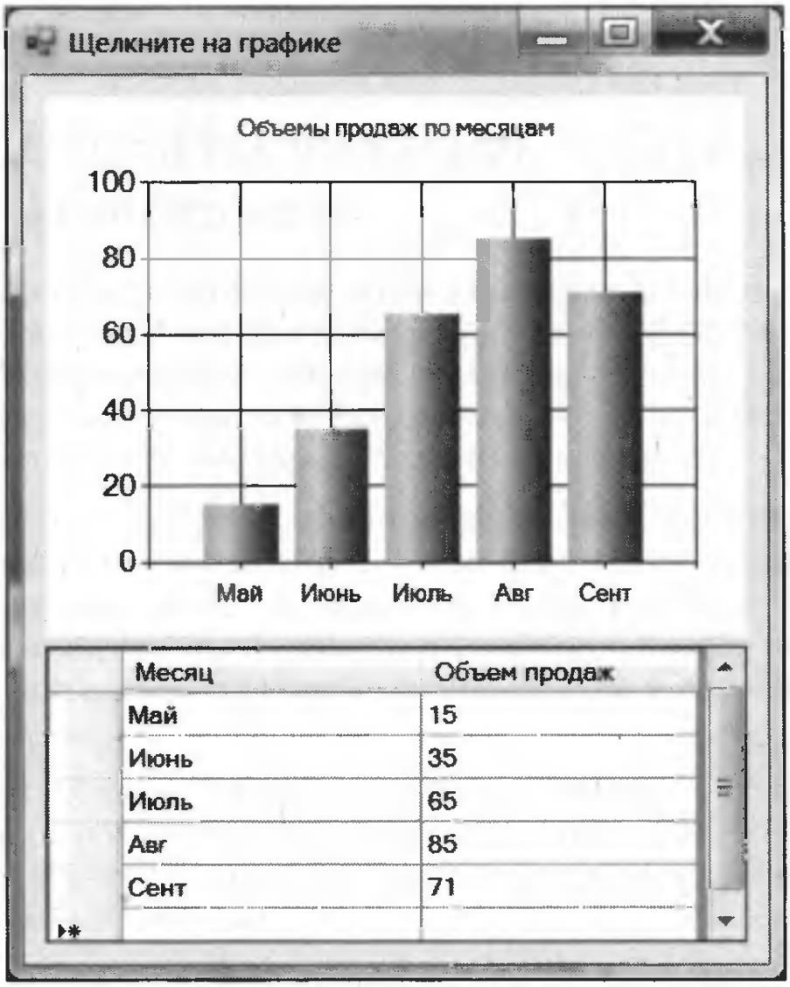


Рис. 13. Проект «Построение диаграммы с помощью элемента Chart»

2. Разработать простейшее приложение GDI+, которое отображает биржевой график изменения продаж некоторого товара (смартфонов) по месяцам в течение года (рис. 14). Названия месяцев для горизонтальной оси поместить в строковый массив Months. Массив целых чисел Sales должен содержать объемы продаж по каждому месяцу, они соответствуют вертикальным ординатам графика. Оба массива должны иметь одинаковую размерность.

При обработке события «щелчок на кнопке Button» следует создать объект Graphics, используя элемент PictureBox. После чего вызвать соответствующие процедуры для рисования координатных осей, сетки из горизонтальных и вертикальных линий и непосредственно эпюра (графика).

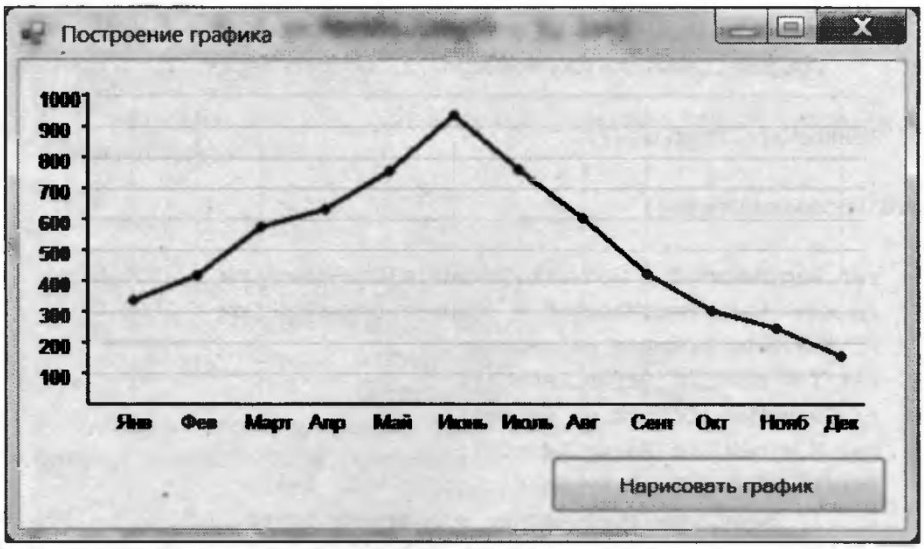


Рис. 14. Проект «График объемов продаж по месяцам»

Программные решения для лабораторной работы № 11:
- задание № 1 (см. Зиборов В.В. – с. 190.).
- задание № 2 (см. Зиборов В.В. – с. 145.).

1. Разработать приложение GDI+, которое демонстрирует экзотический эффект – экранную форму с треугольником прозрачности (рис. 15). Такой треугольник выглядит, как отверстие в форме проекта, через которые видны другие приложения, запущенные в данный момент времени на компьютере.

Для решения задачи следует разработать событие перерисовки элемента управления Form1_Paint (рис. 16). При обработке события перерисовки экранной формы необходимо задать три вершины треугольника, инициализируя массив точек. Для рисования закрашенного многоугольника следует воспользоваться методом FillPolygon. Цвет закрашивания может быть любым, при этом этот же цвет должен быть назначен в качестве прозрачного в свойстве формы TransparencyKey.

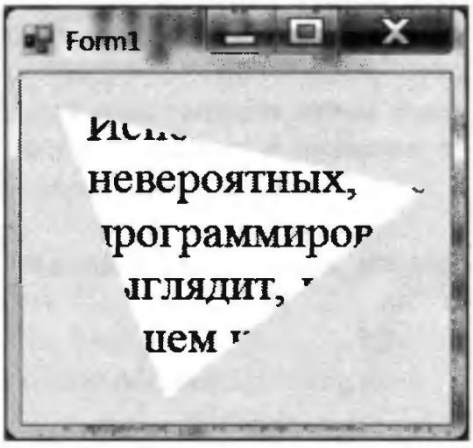


Рис. 15. Экранная форма с треугольником прозрачности внутри

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    // Событие перерисовки экранной формы:
    this.ClientSize = new Size(240, 200);
    // Устанавливаем вершины треугольника:
    var p1 = new Point(20, 20);
    var p2 = new Point(225, 66);
    var p3 = new Point(80, 185);
    // Инициализируем массив точек:
    Point[] Точки = { p1, p2, p3 };
    // Закрашиваем этот треугольник цветом ControlDark:
    e.Graphics.FillPolygon(new SolidBrush(
                        SystemColors.ControlDark), Точки);
    // Цвет ControlDark задаем прозрачным:
    this.TransparencyKey = SystemColors.ControlDark;
}
```

Рис. 16. Событие перерисовки экранной формы

2. Разработать приложение GDI+, которое позволяет пользователю рисовать мышью на форме при нажатой левой (или правой) кнопки мыши (рис. 17). Для этого разработать методы для событий MouseUp и MouseDown. Также разработать метод Refresh для перерисовывания формы при нажатии кнопки Стереть (для очистки формы ее надо закрасить исходным цветом).

Рис. 17. Рисование указателем мыши в форме

3. Разработать приложение GDI+, которое позволяет строить фундаментальные сплайны и сплайны Безье (рис. 18). Для построения сплайна использовать функцию DrawBezier с заданием узловых точек и дополнительных контрольных точек, управляющих кривизной сплайна. Перемещая контрольные точки можно управлять кривизной сплайна. При изменении положения двух контрольных точек можно добиться отрицательного и положительного значений кривизны сплайна.

На экранной форме расположить две узловые точки, а обе управляющие соединить в одну. Разработать метод для перемещения управляющей точки в виде красного прямоугольника для изменения формы сплайна Безье.

Рис. 18. Построение и изменение вида сплайна Безье

Программные решения для лабораторной работы № 12:
- задание № 1 (см. Зиборов В.В. – с. 130.).
- задание № 2 (см. Зиборов В.В. – с. 139.).
- задание № 3 (см. Зиборов В.В. – с. 141.).

Простейший пример использования библиотеки OpenGL в программе на языке программирования Visual C#. Приложение производит отрисовку средствами OpenGL некоторых параметрических поверхностей, например, бутылку Клейна (рис. 19). Процедуры генерации и отрисовки поверхностей содержатся во вспомогательной библиотеке ***Common Classes***. Кроме того, программа использует еще две вспомогательные библиотеки: ***Auxiliary Graphics Library*** (выполняет некоторые служебные графические операции) и ***Auxiliary Math Library*** (набор классов для работы с векторами и матрицами). Данные библиотеки используются во всех последующих примерах.

Отметим, что для работы с API OpenGL используется инструментарий Tao Framework [http://www.taoframework.com]. В частности, для вывода изображения в интерфейс пользователя служит специальный элемент управления ***Tao.Platform.Windows.SimpleOpenGlControl***, который полностью избавляет программиста от выполнения низкоуровневых операций по работе с контекстом рендеринга.

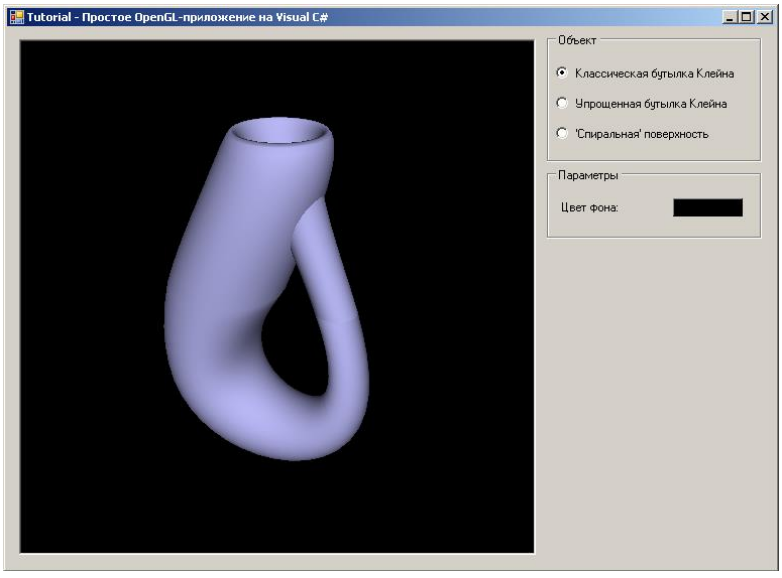


Рис. 19. Главное окно приложения Tutorial - OpenGL Demo

Задание

- Внимательно ознакомьтесь с особенностями работы с API OpenGL с помощью инструментария Tao Framework. Обратите внимание, что для вызова методов OpenGL используются одноименные вызовы в классе ***Tao.OpenGl.Gl*** (все поля и методы данного класса являются статическими, поэтому для работы с ними не требуется создания экземпляра класса). Изучите визуальный элемент управления ***Tao.Platform.Windows.SimpleOpenGlControl***, с помощью которого осуществляется вывод изображения в интерфейс.

- Добавьте свои варианты объектов (это могут быть, например, другие параметрических поверхности – лист Мебиуса, конус, цилиндр или тор) и запрограммируйте их в классе ***Surfaces*** (проект ***Common Classes***).

- Добавьте на сцену несколько (например, два) различных источников света. Поэкспериментируйте со свойствами материала и характеристиками источников.

- Добавьте в функцию отрисовки объектов (класс ***Surfaces***) вычисление текстурных координат (предполагается, что используется двумерная текстура). Текстурные координаты задают способ наложения на объект какого-либо двумерного изображения (в простейшем случае) и представляют собой двумерный вектор, компоненты которого принимают значения в промежутке [0, 1]. Для параметрических поверхностей в качестве текстурных координат можно взять, например, параметрические координаты (с предварительным масштабированием и сдвигом).

- Наложите на объекты двумерные текстуры. Проще всего для этого воспользоваться вспомогательным классом ***Texture2D*** из библиотеки ***Auxiliary Graphics Library***. Статический метод ***Texture2D LoadFromImage(string fileName)*** позволяет создать текстуру практически из всех распространенных графических форматов (BMP, JPEG, GIF, PNG). Обратите внимание, что для работы текстур при инициализации OpenGL (метод ***bool InitOpenGL()***) следует указать команду ***glEnable(GL_TEXTURE_2D)***.

Данное приложение является небольшой модификацией предыдущего с целью использования простой пары вершинного и фрагментного шейдера для процедурного текстурирования. Шейдер генерирует простую процедурную текстуру – кружки определенного цвета и размера на заданном фоне (рис. 20).

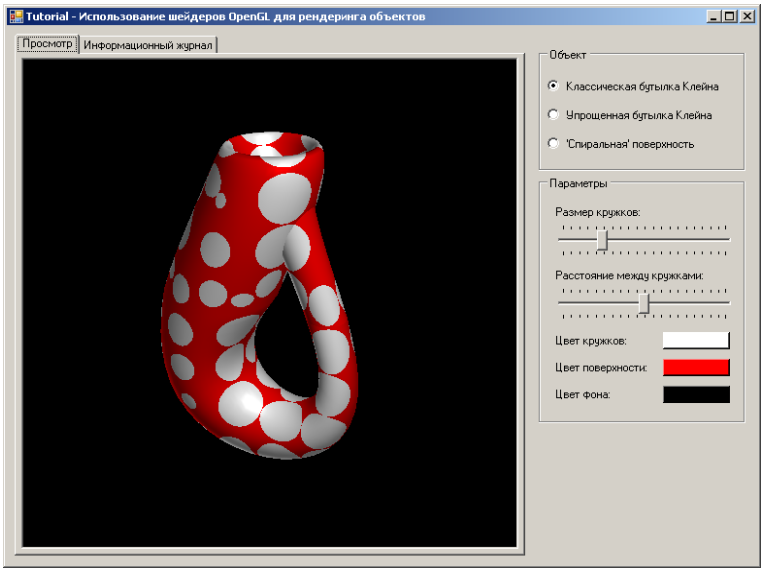


Рис. 20. Главное окно приложения Tutorial - Simple Shader

Задание

- В чем состоит недостаток данного шейдера и как его исправить (обратите внимание на зависимость изображения от ориентации модели)?

- В чем на ваш взгляд состоят преимущества и недостатки процедурных текстур? Какие подходы можно предложить для исправления недостатков?

- Придумайте и запрограммируйте свою процедурную текстуру в шейдере. Текстура может быть как двумерной, так и трехмерной (например, можно взять шахматную доску, полоски, звездочки и т.д.). Постарайтесь добиться того, чтобы текстура была сглажена. Для этого могут использоваться функции типа ***Type smoothstep(float edge0, float edge1, Type x)***, которые обеспечивают плавный переход между двумя значениями (например, цветами).

- Старайтесь не использовать в коде шейдера фиксированные значения параметров. Лучше всего данные параметры вынести в интерфейс шейдера, оформив их в виде ***uniform***-переменных. Добавьте возможность изменения параметров шейдера из интерфейса приложения, используя такие элементы управления, как бегунки (***System.Windows.Forms.TrackBar***), числовые поля ввода (***System.Windows.Forms.NumericUpDown***) и т.п.

Данное приложение дает пример использования шейдеров для обработки изображений (применение некоторых фильтров). В программе реализованы типовые фильтры: прямоугольный сглаживающий фильтр, фильтр выделения границ, гамма-коррекция и приведение к оттенкам заданного цвета (рис. 21).

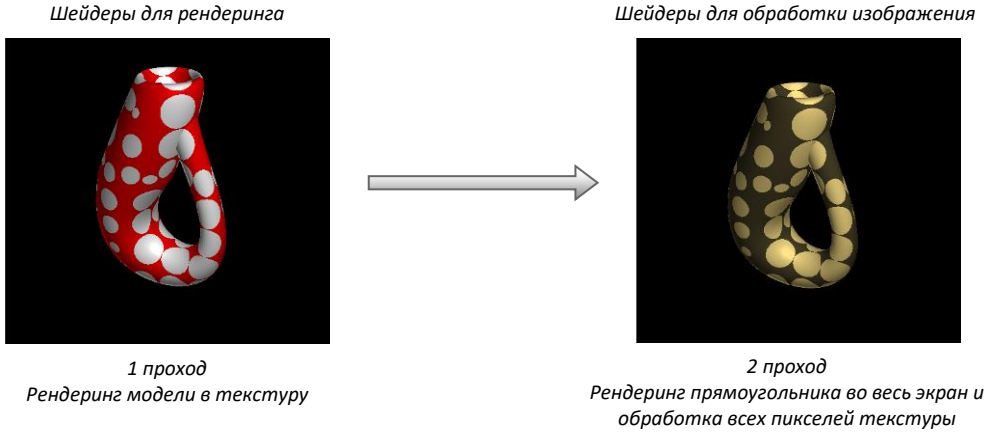


Рис. 21. Иллюстрация техники двухпроходного рендеринга

Для обработки изображения, полученного средствами OpenGL API, используется так называемая техника многопроходного рендеринга (в нашем случае выполняется два прохода). При первом проходе производится рендеринг сцены (в нашем случае это одна из параметрических поверхностей) средствами OpenGL, при этом результат выводится не в главное окно, а в текстуру в памяти видеоадаптера (пользователь не должен видеть промежуточный вариант). При втором проходе осуществляется обработка полученной текстуры, и результат вводится на экран. Для этого следует установить матрицу ортогональной проекции и отобразить прямоугольник во весь размер окна. В процессе рендеринга данного прямоугольника фрагментный шейдер обработает все его пиксели. Таким образом, подав на вход шейдеру полученную ранее текстуру, можно обработать все ее пиксели в соответствии с некоторым фильтром (рис. 22).

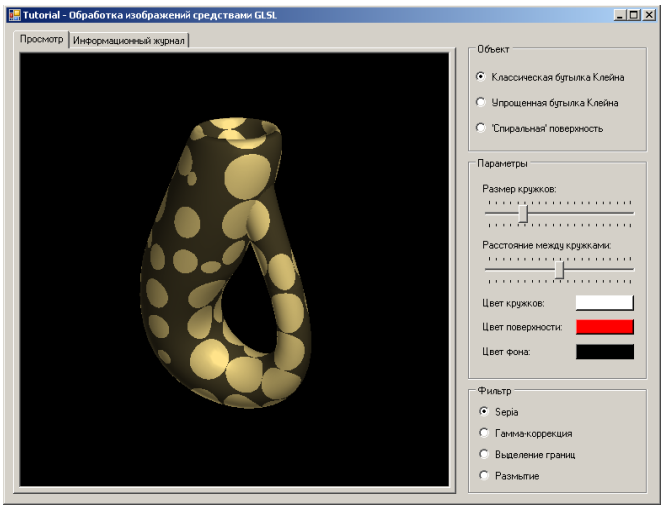


Рис. 22. Главное окно приложения Tutorial - Image Processing

Заметим, что для рендеринга в текстуру доступно несколько вариантов, однако наиболее оптимальный состоит в использовании буфера кадра (Frame Buffer Object – FBO). Вся необходимая функциональность по работе с буфером кадра содержится в классе ***Framebuffer*** (библиотека ***Auxiliary Graphics Library***).

Задание

- Внимательно изучите фильтры, реализованные в данном приложении, и добавьте в них параметры в виде ***uniform***-переменных, которые можно менять из интерфейса приложения. Изучите, как меняется эффект при изменении данных параметров.
- Реализуйте свой собственный фильтр. Можно добавить, например, простейший фильтр инверсии цвета или более сложные фильтры гауссовского сглаживания или изменения контраста. Примеры фильтров на языке C# можно найти в GDI+ Tutorial, приложение Tutorial–Image Processing (они без труда переносятся на GLSL).
- Параметризуйте свой фильтр и предоставьте пользователю возможность изменять его параметры из интерфейса приложения.

Данный пример посвящен изучению шумовой функции, которая играет огромную роль в моделировании различных явлений и материалов. Шейдер использует трехмерную шумовую функцию для вычисления цвета фрагмента, при этом пользователь из интерфейса приложения может изменять такие параметра, как число октав и масштаб шумовой функции (рис. 23).

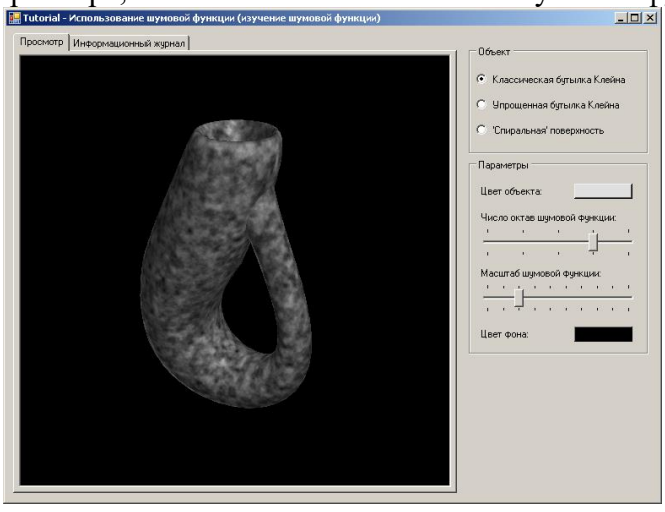


Рис. 23. Главное окно приложения Tutorial - Noise Function

Задание

- Внимательно ознакомьтесь с кодом приложения, поэкспериментируйте с параметрами шумовой функции: число октав, масштаб и др. Подумайте, какие природные явления, минералы и т.д. могут быть смоделированы при помощи шумовой функции.
- Реализуйте шейдер для рендеринга объектов с классическим освещением по модели Фонга (использовалась во всех предыдущих примерах). Однако перед вычислением интенсивности освещения исказите нормаль при помощи шумовой функции (сохраненной в текстуре). Для этого достаточно добавить к правильной нормали вектор шума. Масштаб, число октав и вклад шумовой функции (в искажение нормали) вынесите в интерфейс шейдера, оформив их в виде ***uniform***-переменных. Предоставьте пользователю возможность изменять параметры шейдера из интерфейса приложения. В результате должен получиться эффект «помятости» объекта. При этом внешний вид "помятости" можно будет регулировать из интерфейса.

Данные примеры демонстрируют использование шумовой функции для моделирования дерева и мрамора. Обратите внимание, что данные приложения не используют никаких заранее заготовленных текстур (кроме трехмерной текстуры шума). Все вычисления выполняются во фрагментном шейдере (рис. 24).

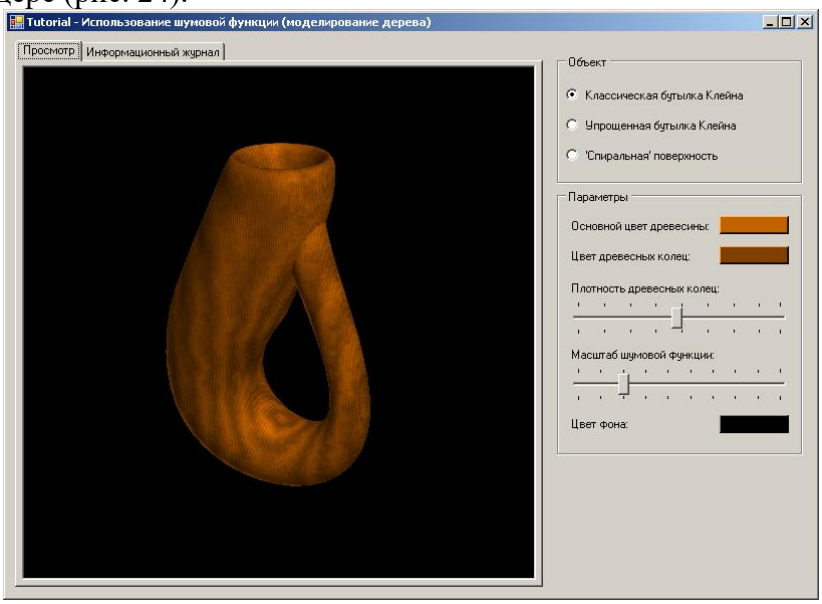


Рис. 24. Главное окно приложения Tutorial - Wood Modeling

Задание

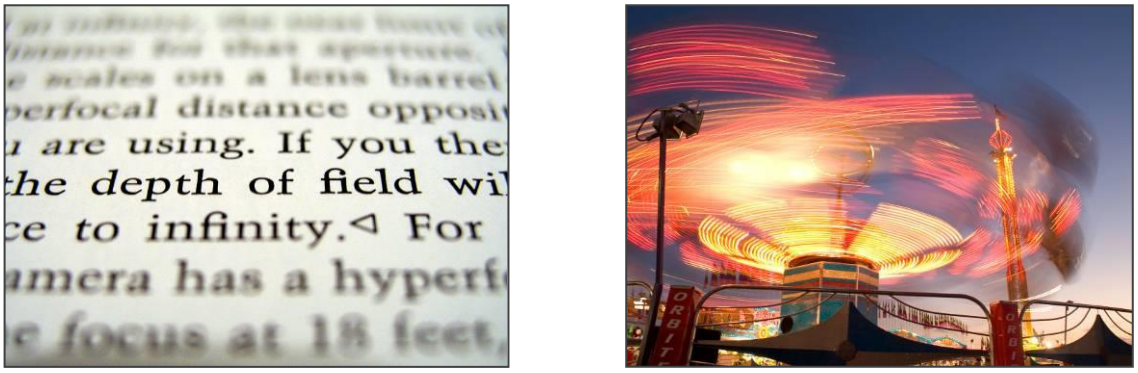
- Изучите данные примеры использования шумовой функции, поэкспериментируйте с параметрами шейдеров. Обратите внимание, что в интерфейс приложения вынесены только основные параметры. Остальные параметры объявлены как константы (с атрибутом ***const***) и могут быть изменены только в исходном коде шейдера.
- Создайте свой собственный эффект на основе шумовой функции. В выборе эффекта могут помочь многочисленные ресурсы Интернет. Можно реализовать, например, простую водную поверхность, эффект облачного неба (описан в [21]), эффект "ржавления" объекта (описан в [21]) и т.д.
- Для того чтобы приложение получилось исследовательским, вынесите основные параметры шейдера в интерфейс пользователя.

Данные примеры знакомят с двумя эффектами, которые сегодня встречаются практически в каждой компьютерной игре – это эффекты "глубины резкости" (Depth of Field) и "размытия движения" (Motion Blur). Оба эффекта имеют под собой реальную почву и могут наблюдаться в повседневной жизни (рис. 25).

Так, например, разглядывая фотографию, мы обнаруживаем, что резкими получаются только те объекты, которые находятся "в фокусе", в то время как близкие объекты и фон плавно

размываются. При рендеринге некоторой сцены на компьютере изображения получается абсолютно резким вне зависимости о того, как расположена камера – процесс синтеза изображения не учитывает особенности реальной оптики. Эффект “глубины резкости” пытается внести в компьютерную графику это явление.

Другое часто встречающееся явление связано с быстрым движением объектов. Напомним, например, быстро вращающееся велосипедное колесо: мы отчетливо видим лишь его обод, в то время как отдельные спицы размываются в “дымку” и становятся неразличимыми. С другой стороны, изображение на экране компьютера воспроизводится недостаточно быстро для естественного возникновения данного эффекта (обычно несколько десятков кадров в секунду или меньше). Таким образом, возникает проблема передачи ощущения от быстро движущихся объектов. Суть эффекта “размытие движения” состоит в искусственном размывании контуров объектов для создания похожих на реальные ощущения.



а) *б*)

Рис. 25. Фотографии, демонстрирующие природу эффектов “глубины резкости” (а) и “размытия движения” (б). Источник – Wikipedia [http://en.wikipedia.org/wiki]

Интересно заметить, что в процессе компьютерной игры или просмотре фильма, вы, вероятно, даже не обратите внимания на данные эффекты. Как и в случае со многими другими побочными эффектами фотографии, вы быстро заметите отсутствие привычных ощущений, в то время как их наличие просто дает вам ощущение реальности, хотя на сам эффект вы даже и не обращаете внимания.

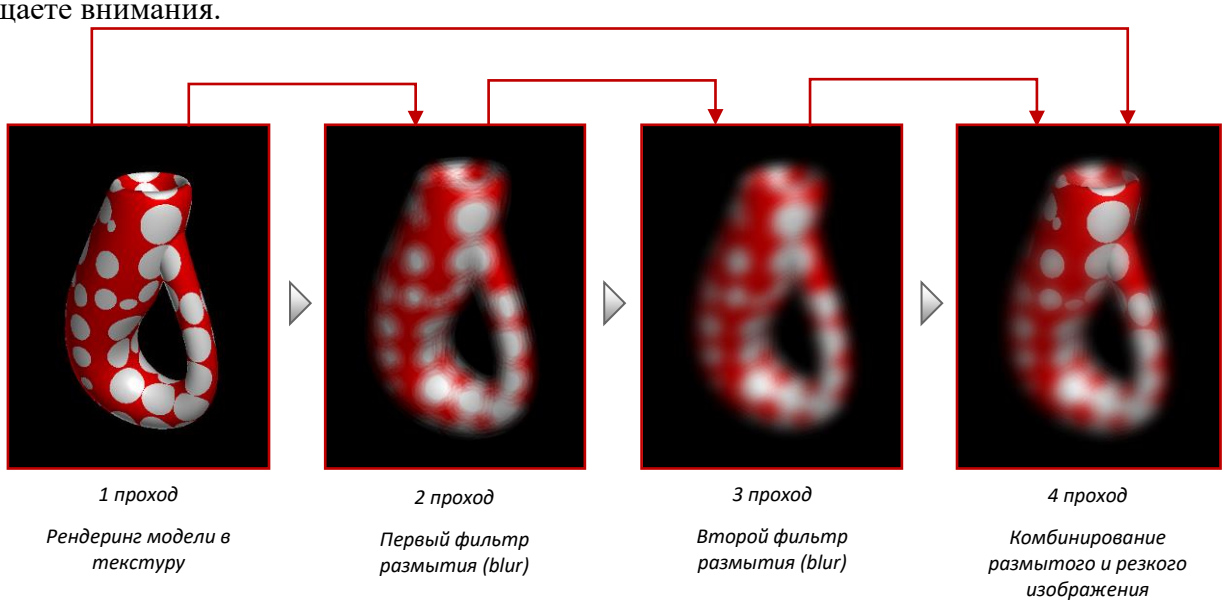


Рис. 26. Процесс формирования эффекта «глубины резкости»

Оба эффекта реализуются совершенно аналогично и используют технику многопроходного рендеринга (multi-pass rendering). Рассмотрим, например, реализацию эффекта “глубины резкости”. Рендеринг каждого кадра осуществляется в четыре прохода. При первом проходе модель визуализируется обычным образом, но результат записывается в

текстуру. Далее следуют два прохода по размытию оригинального изображения (можно ограничиться и одним, однако может пострадать качество результата). При последнем (четвертом) проходе оригинальное изображение и его размытая версия комбинируются на основе данных о глубине каждого пикселя (рис. 27, 28).

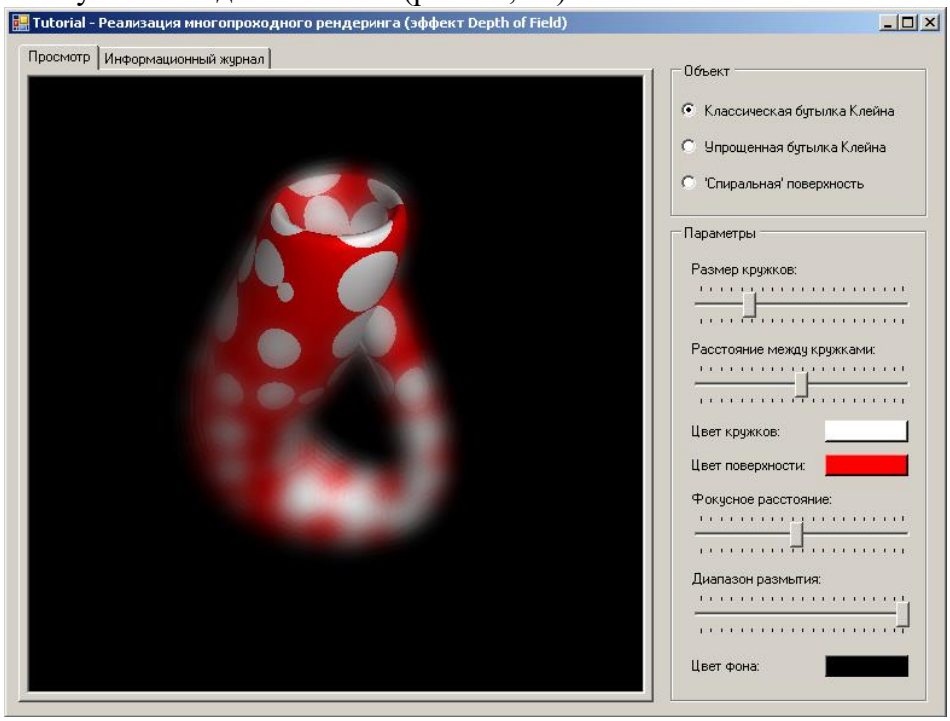


Рис. 27. Главное окно приложения Tutorial - Depth of Field

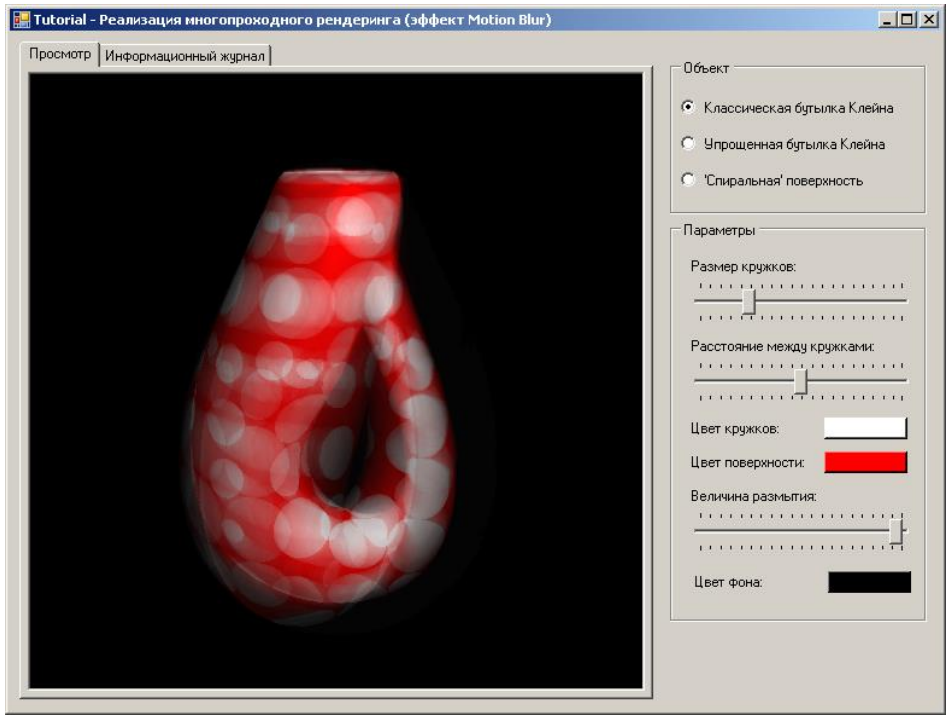


Рис. 28. Главное окно приложения Tutorial - Motion Blur

Задание

- Изучите данные примеры использования буфера кадра. Поменяйте различные параметры шейдера и проследите за изменением эффекта.
- Создайте свой эффект с использованием буфера кадра или модифицируйте один из предложенных (например, добавьте фильтрацию итогового изображения). Различные примеры

эффектов и фильтров можно найти в Tutorial - Image Processing, а для реализации фильтрации потребуется добавить еще один проход.

- Параметризуйте свой шейдер, добавьте возможность изменения его параметров из интерфейса приложения.

Данное приложение дает пример сложного текстурирования (невозможно выполнить стандартными средствами OpenGL API) с помощью нескольких текстур для более реалистичной визуализации планеты Земля. Приложение Tutorial - Earth Planet использует четыре текстуры – текстуру “дневной” и “ночной” планеты, карту отражений, а также трехмерную текстуру шумовой функции.

На освещенную часть планеты (dot(n, l) > 0) накладывается “дневная” текстура, а на затемненную (dot(n, l) < 0) – “ночная”. На границе (где dot(n, l) мало) используется смесь двух текстур для создания эффекта “утра/вечера”. В таком виде пример был описан в [20]. Однако, абсолютно гладкие водоемы и океаны, дающие идеально круглые блики, выглядели крайне нереалистично. Для исправления данного недостатка была применена шумовая функция, с помощью которой искажалась идеальная нормаль к водной поверхности. Информацию о том, где размещены водоемы, а где суша, предоставляет карта отражений земной поверхности.

Приложение Tutorial - Clouds Earth представляет собой развитие предыдущего примера. Оно добавляет анимацию водной поверхности и простейшие движущиеся процедурные облака на основе шумовой функции. Никаких дополнительных текстур не используется; немного усложняется фрагментный шейдер (рис. 29).

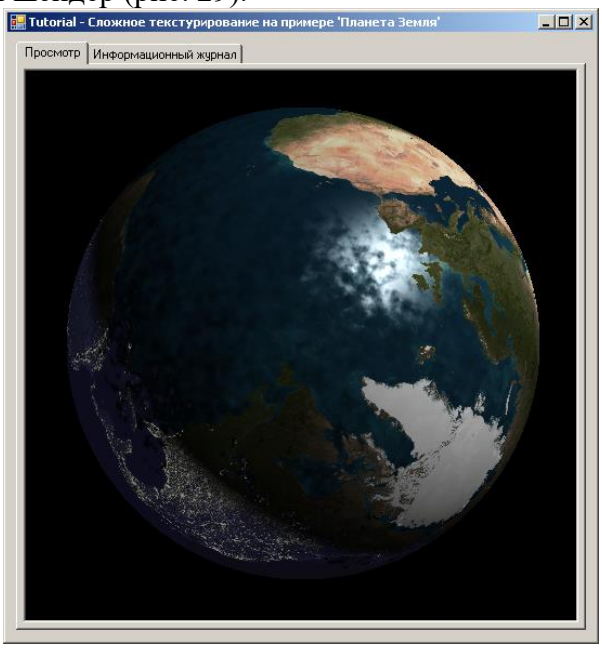


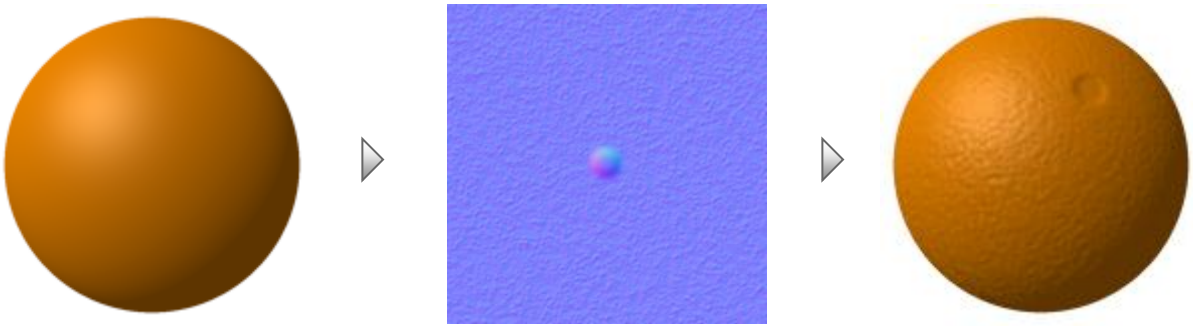
Рис. 29. Главное окно приложения Tutorial - Earth Planet

Задание

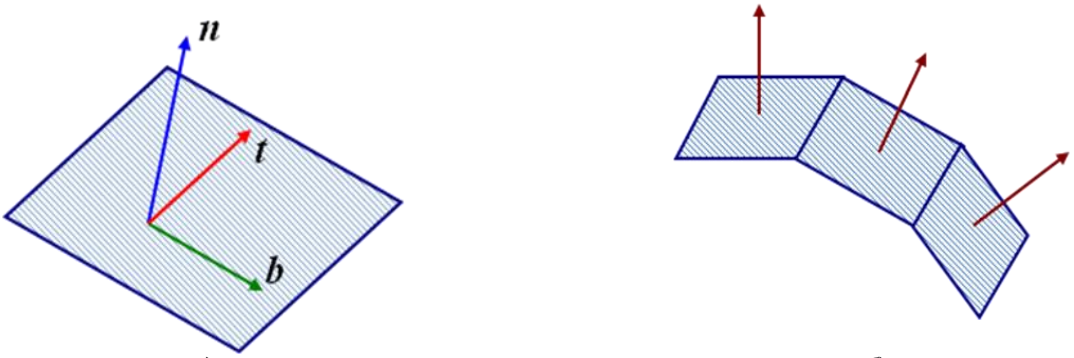
- Наложите на объект дополнительную текстуру – карту облаков (ее просто найти в сети Интернет по запросу “earth clouds map”). Можно попытаться смоделировать облака процедурно с помощью шумовой функции. В Tutorial - Clouds Earth использован простейший подход, который дает не вполне реалистичный результат. С одним из примеров процедурного моделирования облаков можно ознакомиться в [21].
- Сделайте источник света динамическим, чтобы он летал вокруг планеты и освещал ее с различных сторон.
- Оформите важнейшие параметры шейдера в виде ***uniform***-переменных и вынесите их в пользовательский интерфейс.

Данный пример иллюстрирует использование чрезвычайно популярной техники – bump mapping (точнее, одного из ее вариантов – normal mapping). С помощью данного приема можно

значительно улучшить качество изображения, нанеся на объекты микрорельеф и не усложняя при этом геометрию. Идея техники проста: в процессе по-пиксельного расчета освещения bump-текстурой в каждом текселе немного искажается нормаль к объекту таким образом, чтобы получился некоторый узор, который, по существу, является лишь игрой света и тени (рис. 30).



Непосредственное использование карт нормалей затруднено: нормаль задана относительно данной грани и ее неискаженный вектор равен (0, 0, 1) (именно поэтому карты нормалей окрашены преимущественно в синие оттенки).



а) *б*)

Рис. 31. Неискаженный вектор нормали к грани определяется в касательном пространстве (а) и равен (0, 0, 1)

Для расчета освещенности фрагмента необходимо переходить в так называемое касательное пространство (tangent space). В качестве базиса данного пространства выбираются три вектора:

- Касательный вектор t – лежит в плоскости грани;
- Вектор бинормали b – лежит в плоскости грани и ортогонален вектору t;
- Вектор нормали n – ортогонален грани в данной точке.

Для перевода произвольного вектора p в касательное пространство достаточно вычислить три скалярных произведения: $(p \cdot t)$, $(p \cdot b)$, $(p \cdot n)$. В касательном пространстве координаты вектора неискаженной нормали всегда равны (0, 0, 1), поэтому в данном пространстве можно свободно использовать карты нормалей. Остановимся также на получении координат вектора искаженной нормали.

Карта нормалей – это текстура, содержащая закодированные тройками RGB (каналов цвета) единичные нормали к поверхности объектов. Формулы перехода:

$$\begin{aligned} r &= 0.5 \cdot (nx + 1) \\ g &= 0.5 \cdot (ny + 1) \\ b &= 0.5 \cdot (ny + 1) \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} nx &= 2 \cdot r - 1 \\ ny &= 2 \cdot g - 1 \\ nz &= 2 \cdot b - 1 \end{aligned}$$

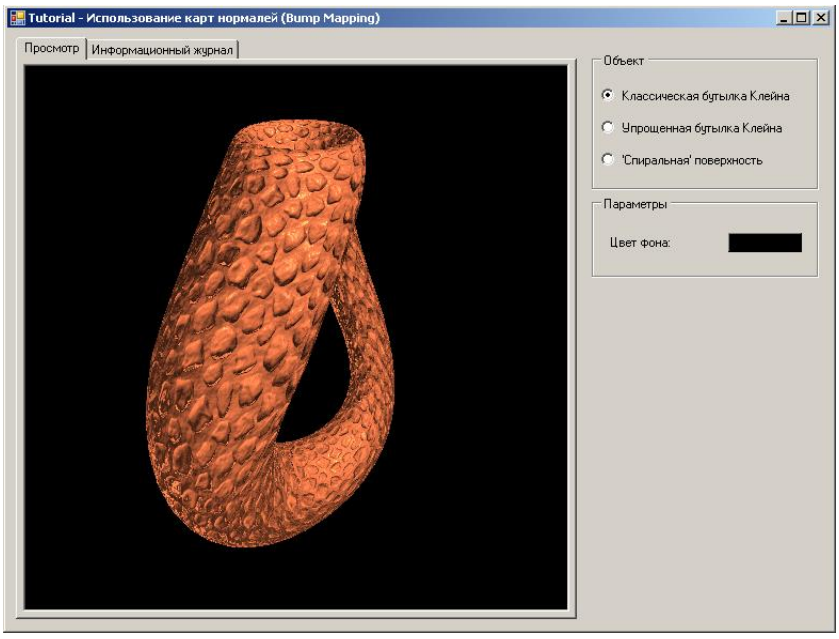


Рис. 32. Главное окно приложения Tutorial - Bump Mapping

Задание

- В данном примере на объект накладывается лишь карта нормалей. Усовершенствуйте его, наложив на объект текстуру и, возможно, карту отражений (использовалась в Tutorial - Earth Planet).
- Сделайте источник света динамическим, чтобы он летал вокруг объекта и освещал его с различных сторон. В этом случае микрорельеф поверхности будет лучше заметен.
- Оформите важнейшие параметры шейдера (например, степень искажения нормалей) в виде ***uniform***-переменных и вынесите их в пользовательский интерфейс.

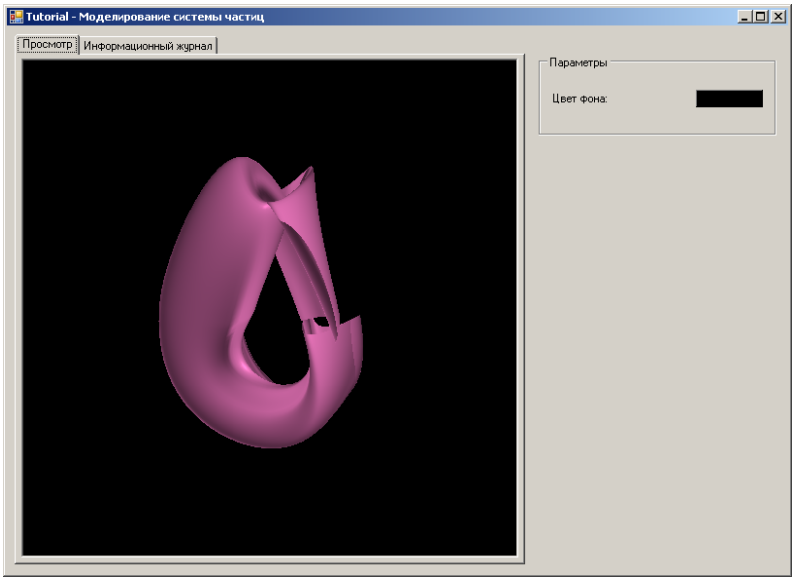


Рис. 33. Главное окно приложения Tutorial - Keyframe Interpolation

Данный пример (см. рис.28) демонстрирует одну из самых распространенных техник анимации – интерполяция между ключевыми кадрами. Данная техника так же известна как “морфинг”. В анимации в качестве ключевых кадров выбираются два варианта некоторой модели (причем вершины должны находиться во взаимно-однозначном соответствии). Например, главный герой в компьютерной игре в начальной точке может иметь нейтральное выражение лица, а в конечной – улыбаться. Далее, между двумя ключевыми кадрами с

помощью интерполяции вставляются дополнительные кадры, и изображение плавно трансформируется от одного кадра к другому.

Задание

- Попробуйте выполнить интерполяцию на основе ключевых кадров для других двух объектов (например, сфера и тор).
- Интерполяция производится на основе некоторой временной функции (например, sin(t)), обычно периодической. Рассмотрите различные варианты таких функций и проследите, как изменится анимация.

Традиционные подходы хорошо подходят для рендеринга гладких и ровных поверхностей, но практически неприменимы для отрисовки нечетких поверхностей – огня, дыма, распыляемой жидкости, фейерверков. Нечеткие объекты не имеют определенной формы и выраженных границ. Попытка их визуализации традиционными подходами приводит к крайне нереалистичному восприятию.

Вначале 80-х гг. Билл Ривз и его коллеги изобрели новый метод для рендеринга нечетких объектов – системы частиц. Данный подход имеет три основных отличия от традиционного рендеринга:

- Объект – это скопление частиц, определяющих объем, а не набор многоугольников, определяющих поверхность;
- Объект всегда считается динамическим: частицы рождаются, меняются и исчезают;
- Объект определен не совсем четко: заданы лишь вероятностные правила возникновения, изменения и исчезновения частиц.

Для упрощения рендеринга системы частиц обычно вводят следующие предположения:

- Частицы не могут перекрываться;
- Частицы не отражают свет, а излучают его.

Каждая частица определяется набором своих атрибутов, в число которых могут входить:

- Начальное положение;
- Начальная скорость;
- Цвет и прозрачность;
- Размер;
- Время рождения или время жизни.

Атрибуты частицы меняются в зависимости от времени в соответствии с физическими законами (влияние гравитации, ветра, трения или других факторов). Таким образом, сложная система декомпозируется на множество маленьких частиц, каждую из которых сравнительно просто смоделировать.

Приложение Tutorial – Particle System дает пример использования системы частиц для моделирования взрыва хлопушки, начиненной конфетти – в короткий промежуток времени появляется множество маленьких ярких кусочков бумаги. Примем следующие предположения о движении кусочков бумаги (рис. 34):

- Кусочки появляются не все сразу, а постепенно;
- Начальные положения частиц лежат на некоторой малой площадке (для простоты она будет прямоугольной, однако лучше генерировать частицы на некотором круге);
- Начальные скорости случайны, но лежат в одном полупространстве (мы используем равномерное распределение, однако вряд ли оно соответствует действительности);
- На частицы влияет сила гравитации, под действием которой они будут опускаться вниз;
- Цвет частиц генерируется произвольным образом и не меняется со временем.

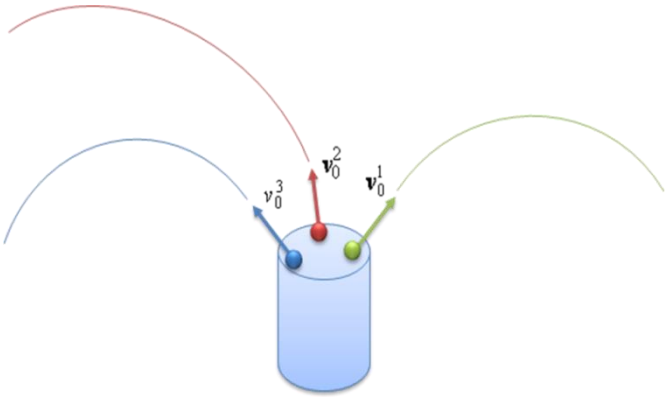


Рис. 34. Кусочки бумаги вылетают из хлопушки с некоторой скоростью и далее движутся под действием силы тяжести

Напомним также уравнение равноускоренного движения тела, которым мы будем пользоваться для расчета положения частицы в вершинном шейдере:

P = P0 + v · t + a · t2 / 2, где

P0 – начальное положение частицы;

v – начальная скорость частицы;

a – ускорение.

Задание

- Изучите данный пример использования системы частиц и подумайте, какие эффекты и объекты могут быть смоделированы таким образом (рис. 35). Измените законы генерации начальных атрибутов частиц: замените прямоугольную площадку на круглую и предложите новое правило получения начальных скоростей (например, для угла отклонения от вертикали, внутри разрешенного угла ±α, принять плотность вероятности пропорциональной cos(πβ/2α)).

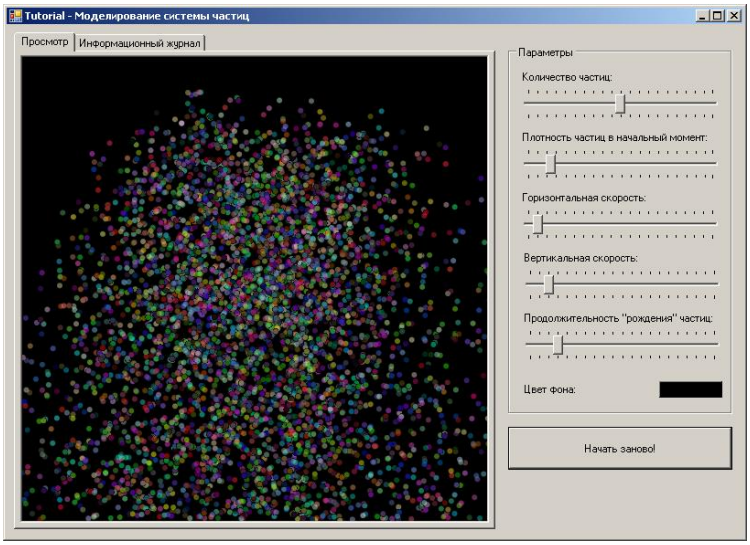


Рис. 35. Главное окно приложения Tutorial - Particle System

- Попробуйте смоделировать какое-либо явление с помощью системы частиц. Например, можно рассмотреть водопад. При этом начальные положения частиц должны лежать на некотором отрезке или ломаной линии (будем считать, что вода падает с некоторого выступа), начальные скорости частиц имеют лишь горизонтальную составляющую, а частицы падают под действием силы тяжести (используется то же самое уравнение движения). Цвет частиц также следует генерировать случайно, однако не во всем множестве возможных цветов, а в некоторой окрестности синего - голубого цвета (при одинаковом цвете частицы будут неразличимы).

Задание

• Расширьте набор доступных объектов для визуализации. Для добавления нового объекта необходимо разработать для него два основных алгоритма: поиск точки пересечения с лучом и расчет нормали в заданной точке (рис. 36).

• Подумайте, каким образом можно использовать трассировку лучей для построения высококачественных изображений неявно заданных поверхностей вида F(x, y, z) = 0. Добавьте новый объект – неявно заданная поверхность (можно ограничиться фиксированным уравнением).

• Реализуйте дополнительные процедурные текстуры: кружочки, звездочки, полосы и т.д. Кроме того, в качестве процедурных текстур могут выступать различные фрактальные изображения: множества Мандельброта и Джулиа, снежинка Коха и узор Серпинского и т.д. Реализуйте процедурную текстуру с одним из данных изображений.

• Добавьте на сцену площадные источники света как совокупность нескольких точечных источников, что позволит получить мягкие (размытые) тени. Подумайте, как образом можно оптимизировать расчеты прямого освещения для площадных источников.

• Реализуйте какой-либо алгоритм сглаживания с целью устранения зазубренности линий и резких цветовых переходов. Подумайте, каким образом можно повысить эффективность таких алгоритмов.

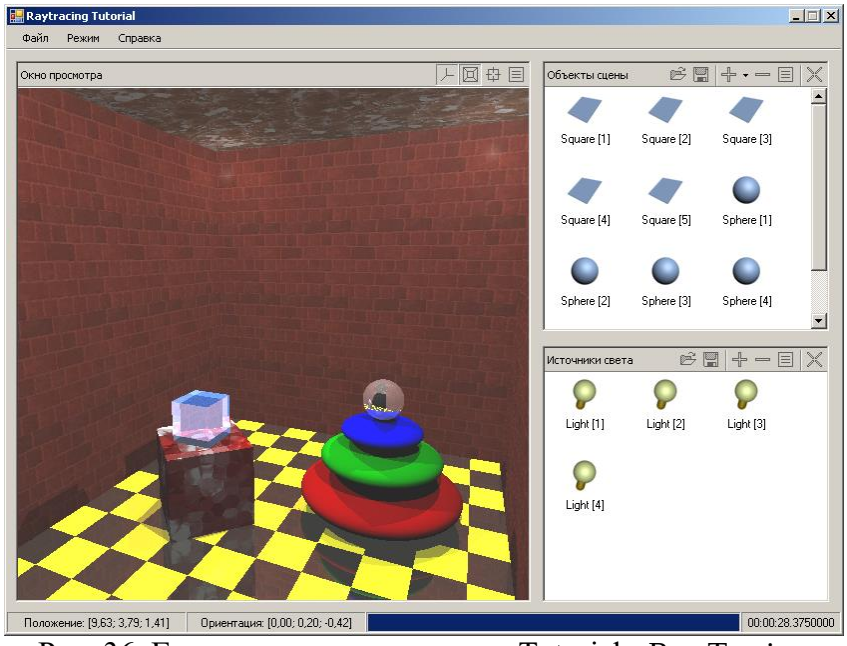


Рис. 36. Главное окно приложения Tutorial - Ray Tracing

Самостоятельная работа

Свободное программное обеспечение

Рассмотрите следующие вопросы

1. Реализация алгоритма Брезенхэма для прямой линии с антиалиасингом и без в форме приложения для тестирования на полигонах.
2. Реализация алгоритма Брезенхэма для окружности с антиалиасингом и без в форме приложения для тестирования.
3. Реализация алгоритма Брезенхэма для эллипса с антиалиасингом и без в форме приложения для тестирования.

Обработка векторной графики

Рассмотрите следующие вопросы

1. Реализация алгоритма Брезенхэма для произвольной дуги с антиалиасингом и без в форме приложения для тестирования.
2. Реализация алгоритма закрашивания линиями для фигуры с цветовой границей в форме приложения для тестирования.

Обработка растровой графики

Рассмотрите следующие вопросы

1. Реализация алгоритма закрашивания линиями для математического описания контура (в форме приложения для тестирования).
2. Реализация редактора для рисования пером и кистью (в форме приложения для тестирования).

Обработка 3D-графики

Рассмотрите следующие вопросы

1. Трехмерное моделирование механизма.
2. Трехмерное моделирование шарнирного робота.
3. Трехмерное моделирование здания.
4. Трехмерное моделирование подъемного крана.
5. Трехмерное моделирование человека.
6. Создание анимации произвольной тематики.

6. Критерии оценивания результатов освоения дисциплины (модуля)

6.1. Оценочные средства и критерии оценивания для текущей аттестации

Виды текущего контроля, предусмотренные рабочей программой дисциплины:

1) устный опрос;
2) выполнение практических заданий лабораторной работы;

1. Требования к устному ответу на вопросы к лабораторному занятию

Ответы студенты должны иллюстрировать конкретными примерами, опираться на теоретическую базу, прослеживать связи между теоретическими и практическими положениями учебной дисциплины, применять теоретические знания к решению вопросов.

Устный ответ предполагает:
- грамотность устной речи;
- убедительность устной речи;
- ясность, точность;
- строгая последовательность, иллюстрация.

Критерии оценки устного ответа

При оценке ответа учитывается:
- полнота и правильность ответа;
- логика изложения;
- степень осознанности и понимания изученного;
- связь теории с практикой.

«Зачтено» ставится, если студент:	- обстоятельно и достаточно полно излагает материал, возможны единичные ошибки; - обнаруживает понимание материала, может обосновать свои суждения, привести примеры; - строит ответ последовательно, возможны отдельные погрешности.
«Незачтено» ставится, если студент:	- обнаружил незнание большей части темы (раздела, вопроса); - при ответе на вопрос искажает его смысл; - излагает материал беспорядочно и неуверенно.

Оценка может быть поставлена студенту как за единовременный ответ, так и за ответ, рассредоточенный во времени, т.е. за сумму ответов, данных в процессе занятий.

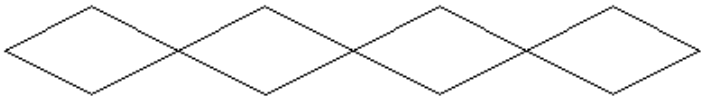
2. Требования к выполнению практических заданий лабораторной работы

Практическое задание лабораторной работы выполняется в письменном (печатном) виде. Это вид учебной работы студента по аналитической обработке информации, принятию самостоятельных решений, инициированию творческих идей.

Примеры практических заданий лабораторной работы

Практическое задание 1 Изобразить на экране цилиндр, выделив пунктиром невидимые линии. Вывести на экран формулу для вычисления объема цилиндра.

Изобразить, используя оператор цикла:



Показатели и критерии оценки задания:

полнота выполнения задания – от 0 до 3 баллов;

правильность выполнения задания (технологически) – от 0 до 3 баллов;

точность расчётов / логичность рассуждений – от 0 до 3 баллов;

аккуратность выполнения – от 0 до 3 баллов.

Шкала оценки: 0 – требование не выполнено; 1 – требование выполнено частично; 2 – требование выполнено, но есть недочёты; 3 – требование выполнено.

«зачтено» – 9 баллов и более;

«не зачтено» – менее 9 баллов.

Для получения оценки «зачтено» по выполнению практических заданий лабораторной работы студент должен получить оценку зачтено по каждому выполнению практического задания лабораторной работы из п.5 данной программы.

6.2. Оценочные средства и критерии оценивания для промежуточной аттестации

Критерий получения зачета

Зачет выставляется по результатам работы студента в течение семестра согласно Положению о текущем контроле успеваемости и промежуточной аттестации студентов в федеральном государственном бюджетном образовательном учреждении высшего образования «Смоленский государственный университет» (утверждено приказом и.о. ректора № 01-113 от 26.09.2019; внесены дополнения приказом ректора № 01-48 от 30.04.2020).

Для получения зачета студент должен:

- уметь отвечать на теоретические вопросы, рассмотренные в самостоятельной работе;
- уметь решать задачи, предложенные на лабораторных занятиях.

Шкала оценивания навыков для получения зачета:

Количество лабораторных работ за которые получено «зачтено»	Оценка
3-4	«Зачтено»
Менее 4	«Не зачтено»

7. Перечень основной и дополнительной учебной литературы

7.1. Основная литература

1. Подбельский, В. В. Программирование. Базовый курс C# : учебник для вузов / В. В. Подбельский. — Москва : Издательство Юрайт, 2020. — 369 с. — (Высшее образование). — ISBN 978-5-534-10616-9. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: http://www.biblio-online.ru/bcode/450868 (дата обращения: 10.03.2021).
2. Зыков, С. В. Программирование. Объектно-ориентированный подход : учебник и практикум для вузов / С. В. Зыков. — Москва : Издательство Юрайт, 2020. — 155 с. —

(Высшее образование). — ISBN 978-5-534-00850-0. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: http://www.biblio-online.ru/bcode/451488 (дата обращения: 10.03.2021).

7.2. Дополнительная литература

1. Инженерная 3d-компьютерная графика в 2 т. Том 1 : учебник и практикум для академического бакалавриата / А. Л. Хейфец, А. Н. Логиновский, И. В. Буторина, В. Н. Васильева. — 3-е изд., перераб. и доп. — М. : Издательство Юрайт, 2017. — 328 с. — (Серия : Бакалавр. Академический курс). — ISBN 978-5-534-02957-4. — Режим доступа : www.biblio-online.ru/book/35643B27-D91B-488F-8E88-7026A126A74D.
2. Инженерная 3d-компьютерная графика в 2 т. Том 2 : учебник и практикум для академического бакалавриата / А. Л. Хейфец, А. Н. Логиновский, И. В. Буторина, В. Н. Васильева. — 3-е изд., перераб. и доп. — М. : Издательство Юрайт, 2017. — 279 с. — (Серия : Бакалавр. Академический курс). — ISBN 978-5-534-02959-8. — Режим доступа : www.biblio-online.ru/book/9ED0809C-145C-47A3-8DB0-2A79F21CE056.
3. Боресков, А. В. Компьютерная графика : учебник и практикум для прикладного бакалавриата / А. В. Боресков, Е. В. Шикин. — М. : Издательство Юрайт, 2018. — 219 с. — (Серия : Бакалавр. Прикладной курс). — ISBN 978-5-9916-5468-5. — Режим доступа : www.biblio-online.ru/book/D39797BE-488C-4EC5-AFE8-F60AE1B9C750.

7.3. Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. Центр дистанционного обучения СмолГУ (cdo.smolgu.ru). - https://cdo.smolgu.ru/
2. Национальный открытый университет (intuit.ru). - https://intuit.ru/
3. Национальная платформа открытого образования (opened.ru). - https://openedu.ru/

8. Материально-техническое обеспечение

Перечень материально-технического обеспечения, необходимого для реализации курса, включает в себя лабораторию, оснащенную компьютерной техникой с возможностью подключения к сети "Интернет", проектором и интерактивной доской ауд.224 на 12 посадочных мест и 6 парт (12 посадочных мест).

Помещение для самостоятельной работы обучающихся оснащено компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду университета, ауд 224. на 12 посадочных мест.

9. Программное обеспечение

1. Система поддержки MS Visual Studio (msdn.ru)
2. Система дистанционного обучения СмолГУ (moodle.smolgu.ru).